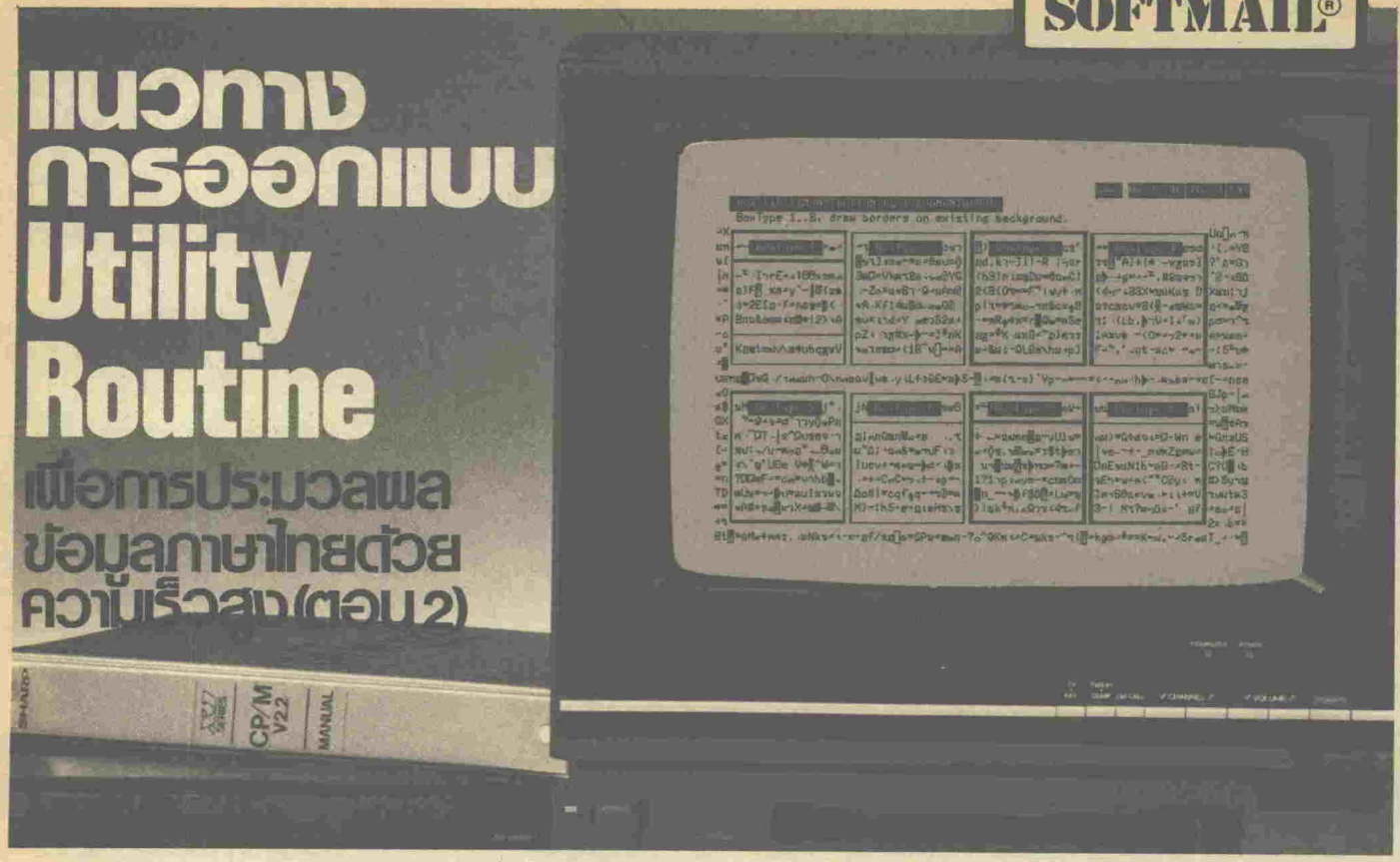
แนวทางการออกแบบ Utility Routine เพื่อการประมวลผลข้อมูลภาษาไทยด้วยความเร็วสูง (ตอน 2)



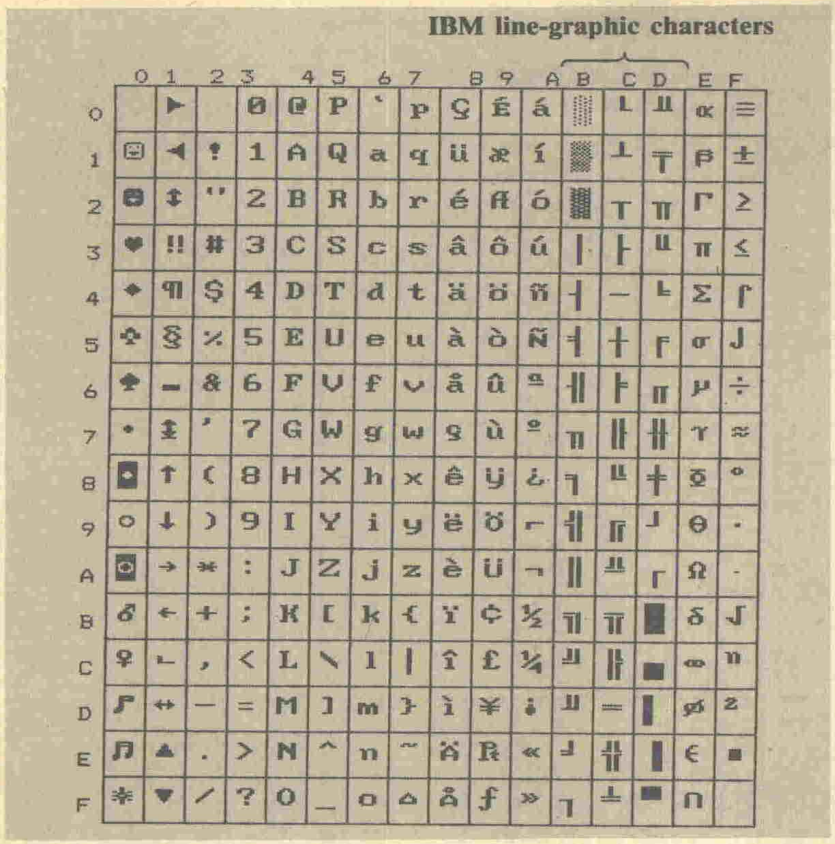
ทวีศักดิ์ กออนันตกูล

โปรแกรมกรอบ ๆ

ในตอนที่ 1 ผู้เขียนได้กล่าวถึงแนวคิดของการนำโปรแกรมเก่ามาใช้งานซ้ำ (reusable software) เพื่อช่วยในการพัฒนาโปรแกรมใหม่ ๆ โดยอาศัยผลงานเก่าที่เคยทำไว้ดี พร้อมกับเสนอการพัฒนา Utility routine สำหรับการประมวลผลข้อมูลภาษาไทย ผู้เขียนได้ยกตัวอย่างพื้นฐานการแสดงตัวอักษรบนจอภาพโดยวิธีการเขียนลงบน VRAM โดยตรง เขียนเป็น utility routine ไว้ 3 เรื่อง เรียกชื่อว่า Define-Vram PutFast และ WriteFast พร้อมกับโปรแกรมทดสอบชื่อ EX1 ให้ผู้อ่านที่สนใจนำไปทดลองด้วยตนเอง เพื่อพบว่า routine เหล่านี้ทำงานเร็วกว่าการเขียนโปรแกรมตามปกติประมาณ 5-10 เท่า

ในตอน 2 นี้เราจะยังไม่เข้าสู่เรื่องของตัว routine เกี่ยวกับภาษาไทยทันที แต่จะแสดงให้เห็นขั้นตอนของการพัฒนาที่ฝรั่งเรียกกันว่า "Stepwise Refinement" หรือการปรับปรุงแบบก้าวสั้น ๆ โดยจะยกตัวอย่าง routine เกี่ยวกับการวาดกรอบสี่เหลี่ยมบนจอไอบีเอ็ม พีซีให้เราสามารถเขียนโปรแกรมขึ้นมาเพื่อใช้เองหรือให้เพื่อนใช้งาน แต่เมื่อพบว่าเป็นที่นิยมกัน การเพิ่มความสามารถให้เก่งขึ้นและเร็วขึ้นในเวลาเดียวกันจะเป็นประโยชน์ต่อทุกคน เพราะเมื่อมีการใช้บ่อยครั้งแล้ว หากโปรแกรมเร็วขึ้นเพียง 10-20% แต่เกิดขึ้นกับทีมโปรแกรมเมอร์ทั้งทีมในหน่วยงาน ผลงานเช่นนี้ย่อมมีค่ามากกว่า 20% สำหรับผู้เขียนเพียงคนเดียวแน่นอน เพราะไม่ใช่แต่ผู้เขียนเท่านั้นที่ได้ประโยชน์ คนอื่น ๆ อีก 5 คน (หรือ 10 คน หรือ 100 คน) ก็ได้ประโยชน์ด้วย ตัวอย่างต่าง ๆ ที่ปรากฏในบทความนี้สามารถนำไปทดลองกับไอบีเอ็ม พีซี หรือคอมแพตทิเบิ้ลทั่วไปที่ใช้การ์ดแสดงผลในลักษณะภาษาอังกฤษล้วน (คือใช้ฟอนต์ภาษาอังกฤษและตีเส้น) ส่วนไทยการ์ดที่ไม่สามารถตีเส้นตีกรอบเมื่อเล่นกับโปรแกรมเหล่านี้แล้วจะออกมาไม่เป็นเส้นแต่กลายเป็นตัวภาษาไทยแทน ทั้งนี้ยกเว้นไทยการ์ดของเทลบิซรุ่น 4FG และ 8FG ซึ่งสามารถแสดงผลเป็นตัวอักษรอังกฤษ-ไทย และตีเส้นแบบไอบีเอ็มแท้ได้พร้อม ๆ กัน วัตถุประสงค์ของตัวอย่างต่าง ๆ ที่นำมาเสนอในตอนนี้ประกอบด้วย

1. ศึกษาแนวทางการวาดกรอบสี่เหลี่ยมบนจอไอบีเอ็ม พีซี
2. ปรับปรุงความสามารถของโปรแกรมให้เก่งขึ้น (คือทำงานได้กว้างขวางขึ้น)
3. ปรับปรุงให้โปรแกรมทำงานเร็วขึ้น โดยเรียกใช้ utility routine ซึ่งเคยเสนอไว้ในตอนที่ 1 ได้แก่ PutFast และ WriteFast ซึ่งในฉบับนี้จะลง Listing ของบทความในฉบับก่อนด้วย (รูปที่ 7 และ 12)
4. แสดงให้เห็นตัวอย่างการเรียกใช้ utility routine ที่สามารถตีกรอบได้ แต่เขียนด้วยภาษา assembly เพื่อให้ทำงานเร็วขึ้นและสามารถเรียกใช้งาน



รูปที่ 1 ตารางรหัสตัวอักษรของไอบีเอ็ม พีซี มีตัวอักษรสำหรับใช้ตีกรอบตีตารางในช่วง BØ จนถึง DF

ได้จากภาษาสูง เช่น Pascal หรือ C โดยสะดวก ผู้อ่านสามารถใช้ตัวอย่างนี้เป็นแนวทางที่จะทำความเข้าใจวิธีการเรียกใช้ utility routine อื่น ๆ อีกมากมายที่ใช้กับคอมพิวเตอร์ในเมืองไทยได้

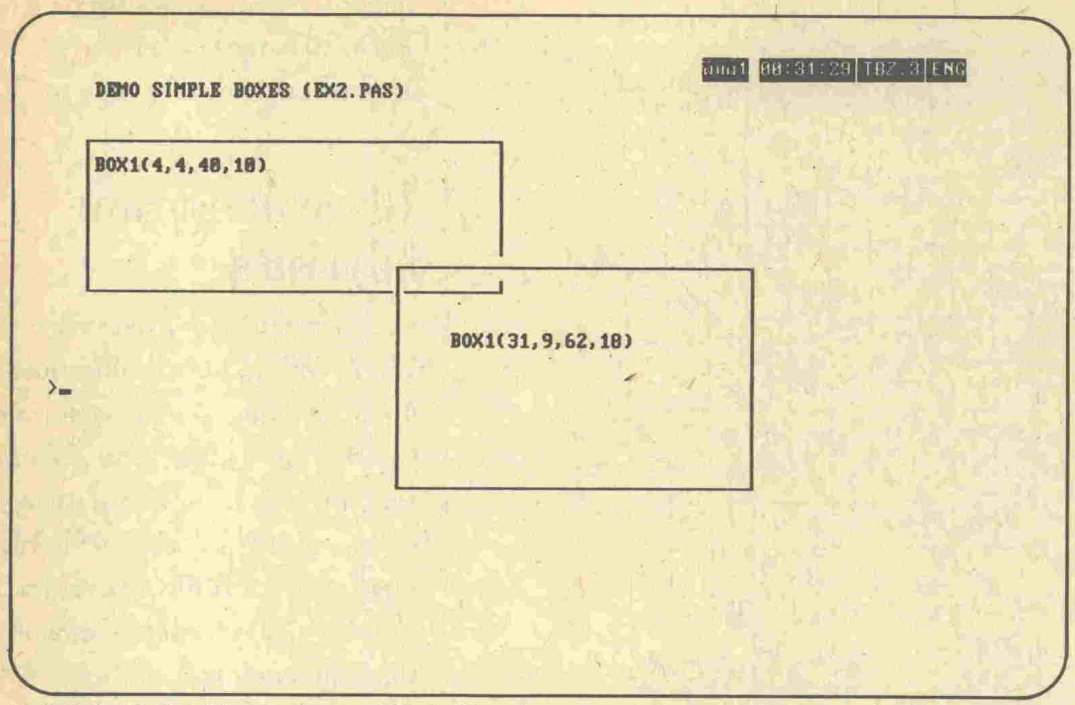
ระบบภาษาไทยที่ผู้เขียนและคณะได้พัฒนาขึ้นใช้ภายในสถาบันประมวลข้อมูลเพื่อการศึกษาและการพัฒนา มหาวิทยาลัยธรรมศาสตร์ อาศัยวิธีการออกแบบ Utility Routine ตามสภาพการเขียนโปรแกรมใช้งาน และเมื่อพบว่าหน้าที่ใดก็ตามที่ใช้งานบ่อย ๆ ไม่ว่าจะเป็นด้านภาษาไทยหรือทั่ว ๆ ไป ส่วนนั้นจะได้รับการปรับปรุงให้เร็วขึ้นและสั้นลง และกลายเป็น service function ของระบบภาษาไทยในที่สุด ฟังก์ชันเหล่านี้โปรแกรมเมอร์สามารถเรียกใช้งานจากภาษาสูงได้โดยสะดวก

โปรแกรมวาดกรอบอย่างง่าย ๆ

เครื่องไมโครคอมพิวเตอร์ตระกูลไอบีเอ็ม พีซี มีตัวอักษรชุดพิเศษที่เรียกกันว่า IBM line-graphic characters (ดูรูปที่ 1) ซึ่งเป็นตัวอักษรที่ต่อเติมจากรหัส ASCII ปกติ ตัวอักษรลากเส้นต่าง ๆ เหล่านี้เป็นที่นิยมใช้ในโปรแกรมสมัยใหม่ที่มักจะแสดงข้อความเป็นช่องหน้าต่างสี่เหลี่ยม โผล่มาตามที่ต่าง ๆ แล้วแต่เหตุการณ์ ในกรณีที่เราต้องการเขียนโปรแกรมขึ้นมาเพื่อวาดกรอบสี่เหลี่ยมแสดงข้อความบ้างก็ย่อมทำได้ไม่ยากนัก ในรูปที่ 2 แสดงโปรแกรม

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  12:28:00                  Listing of EX2.PAS

    1: Program BoxExample;
    2:
    3: { This program illustrate a Standard Pascal way of
    4:   drawing a rectangular box on an IBM-PC screen }
    5:
    6: { ============================ PROCEDURE BOX1 ======= }
    7:   Procedure Box1 (x0,y0,x1,y1 : integer);
    8:   var  i : integer;
    9:   begin
   10:      gotoXY(x0,y0);    write(chr(218));
   11:      for i := succ(x0) to  pred(x1) do write(chr(196));
   12:      write(chr(191));
   13:      for i := succ(y0) to pred(y1) do
   14:         begin
   15:         gotoXY(x0,i);  write(chr(179));
   16:         gotoXY(x1,i);  write(chr(179));
   17:         end;
   18:      gotoXY(x0,y1);  write(chr(192));
   19:      for i := succ(x0) to pred(x1) do write(chr(196));
   20:      write(chr(217))
   21:   end;
   22:
   23: { ========================= END PROCEDURE BOX1 ======= }
   24:
   25:
   26: begin  { Example Main program }
   27:   ClrScr;
   28:   TextBackground(1);
   29:   GotoXY(4,2);
   30:   Writeln(' DEMO SIMPLE BOXES (EX2.PAS) ');
   31:   Box1(4,4,40,10); GotoXY(5,5); Writeln('BOX1(4,4,40,10)');
   32:   Box1(31,9,62,18); GotoXY(36,12); Writeln('BOX1(31,9,62,18)')
   33: end.
   34:
```

รูปที่ 2 โปรแกรมวาดกรอบอย่างง่าย ๆ

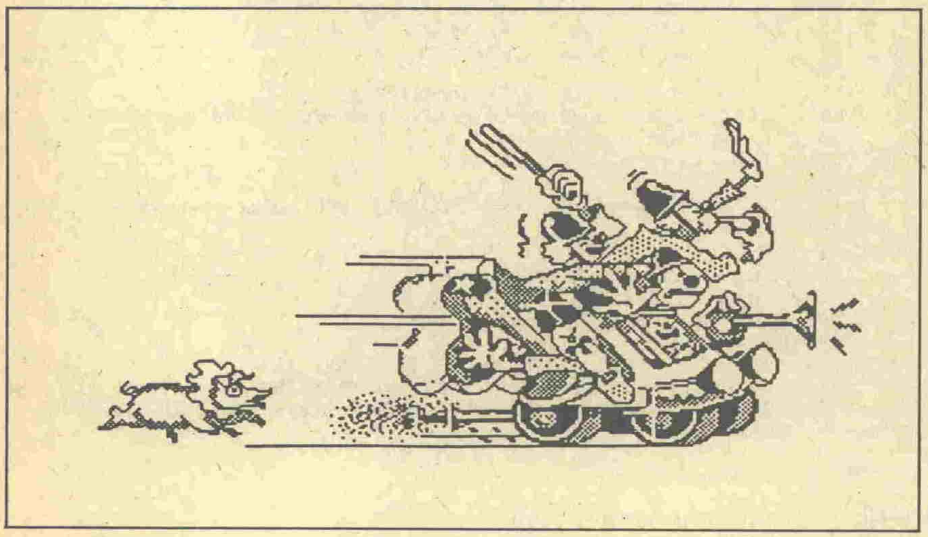


รูปที่ 3
ผลลัพธ์จากโปรแกรมวาดกรอบอย่างง่าย ๆ

EX2.PAS ซึ่งมีการวาดสี่เหลี่ยมโดยอาศัย Procedure Box1 (x0,y0,x1,y1) วาดสี่เหลี่ยมซึ่งมีมุมซ้ายบนอยู่ที่ตำแหน่งตัวอักษรคอลัมน์ x0 บรรทัดที่ y0 และมุมขวาล่างอยู่ที่คอลัมน์ x1 บรรทัดที่ y1 เมื่อทดลองรันโปรแกรม EX2 จะได้ผลบนจอภาพตามรูปที่ 3

Procedure Box1 มีหลักการทำงานง่าย ๆ ดังนี้

- ไปที่ตำแหน่ง x0, y0 แล้ววาดสัญลักษณ์มุมซ้ายบน (┌)
- ลากเส้นนอนจากจุดถัดไป จนถึง (x1-1), y0
- จากนั้นวาดสัญลักษณ์มุมขวาบน (┐)
- ลากเส้นดิ่งลงมา 2 เส้น จากตำแหน่ง x0, (y0+1) กับ x1, (y0+1) ลงมาจนถึง x0, (y1−1) และ x1, (y1−1) ตามลำดับ
- ไปที่ x0, y1 แล้ววาดสัญลักษณ์มุมซ้ายล่าง (└)
- ลากเส้นนอนต่อไปจนถึง (x1-1), y1
- ที่ x1, y1 วาดสัญลักษณ์มุมขวาล่าง (┘)



ถ้าเครื่องของท่านวาดกรอบออกมาเป็นตัวอักษรไทย ก็แปลว่ากำลังใช้ไทยการ์ดอยู่ อาจมีสวิตช์โยกท้ายเครื่องเพื่อเลือกชุดตัวอักษรระหว่างอังกฤษ/ไทย และอังกฤษ/ตีเส้น ถ้ามีสวิตช์นี้อยู่ก็ควรจะไปโยกดูเพื่อจะได้มองเห็นกรอบ ถ้าไม่สามารถเปลี่ยนตัวไทยเป็นตีเส้นได้ ท่านอาจจะลองแก้รหัสตัวอักษรที่ใช้ตีกรอบจาก line graphic มาเป็นตัวภาษาอังกฤษธรรมดา เช่น / ¦ - \

โปรแกรมวาดกรอบหลายรูปแบบ

ตัวอย่าง EX2 แสดงให้เห็นถึงแนวการพัฒนาโปรซีเจอร์วาดกรอบอย่าง

ง่าย ๆ คือมีแบบเดียวแต่มีขนาดต่าง ๆ กันได้ เขียนง่าย อ่านแล้วเข้าใจง่าย แต่เมื่อใช้งานไปสักพักเราอาจจะพบว่าน่าจะทำอะไรได้มากกว่านี้โดยไม่จำเป็นต้องเพิ่มความยาวของโปรแกรมมากนัก ตัวอย่างเช่น เมื่อดูโปรซีเจอร์ Box1 อย่างใกล้ชิด เราอาจจะพบว่าสัญลักษณ์ตีเส้นนอน เส้นดิ่ง มุมซ้ายบน ขวาบน มุมซ้ายล่างและขวาล่าง สามารถผสมได้ว่าจะให้ขอบไหนเป็นเส้นเดี่ยว ขอบไหนเป็นเส้นคู่ ดังนั้นกรอบที่จะวาดก็สามารถทำได้หลายแบบ เช่น กรอบเดี่ยว กรอบเส้นคู่ กรอบคู่ประสมเดี่ยว เป็นต้น

ในกรณีเพิ่มความสามารถของโปรแกรม ถ้าได้ศึกษาถึงความเป็นไปได้ทั้งการแก้ไขอัลกอริทึม หรือโครงสร้างข้อมูลอย่างใกล้ชิด บ่อยครั้งเราพบว่าสามารถเขียนโปรแกรมให้ทำงานเก่งขึ้นโดยการแก้ไขโครงสร้างข้อมูลเพียงอย่างเดียว โดยไม่ต้องแก้อัลกอริทึมเลย สมดังคำกล่าวของศาสตราจารย์ Niklaus Wirth ที่บอกว่า **"อัลกอริทึม + โครงสร้างข้อมูล = โปรแกรม!"**

ตัวอย่าง EX3.PAS (รูปที่ 4) ก็เช่นเดียวกัน ผู้เขียนได้สร้างโปรซีเจอร์ชื่อ Box2 ขึ้นมาโดยอาศัยเค้าโครงจาก Box1 ทั้งหมด แต่ได้แก้ไขวิธีการกำหนดค่าตัวอักษรสำหรับวาดให้กลายเป็นอะเรย์ไป ตัวอย่างเช่น การวาดมุมขวาบนในบรรทัดที่ 12 รูปที่ 2 **Write(chr(191));** กลายเป็นบรรทัดที่ 30 รูปที่ 4 ซึ่งเขียนว่า **write (TopRight [BoxType]);** ทั้งนี้ได้มีการนิยามชื่ออะเรย์ตัวอักษรวาดกรอบไว้ 7 อะเรย์ในบรรทัดที่ 13-21 (รูปที่ 4)

โปรซีเจอร์ Box2 มีพารามิเตอร์ชื่อ BoxType เพิ่มขึ้นมาอีก 1 ชื่อ โดยมีข้อตกลงหรือสเปกว่า BoxType ต้องมีค่าอยู่ระหว่าง 1 ถึง 4 เท่านั้น มิฉะนั้นแล้วจะวาดเป็นกรอบเดี่ยวให้แทน ตัว BoxType นี้เองจะเป็นกุญแจสำคัญในการเลือกชุดของสัญลักษณ์วาดรูปจากอะเรย์ต่าง ๆ ของแถมที่ได้จากการใช้อะเรย์ที่มีชื่อชัดเจน เช่น Topleft หรือ TopRight ก็คือ ทำให้โปรแกรมอ่านแล้วเข้าใจง่ายกว่าเดิมด้วยซ้ำ ในกรณีของโปรแกรม EX3 ความหมายของ BoxType เห็นได้ชัดเจนจากรูปที่ 5 ซึ่งเป็นผลของการรันโปรแกรม EX3 โดยสรุปแล้วก็คือ

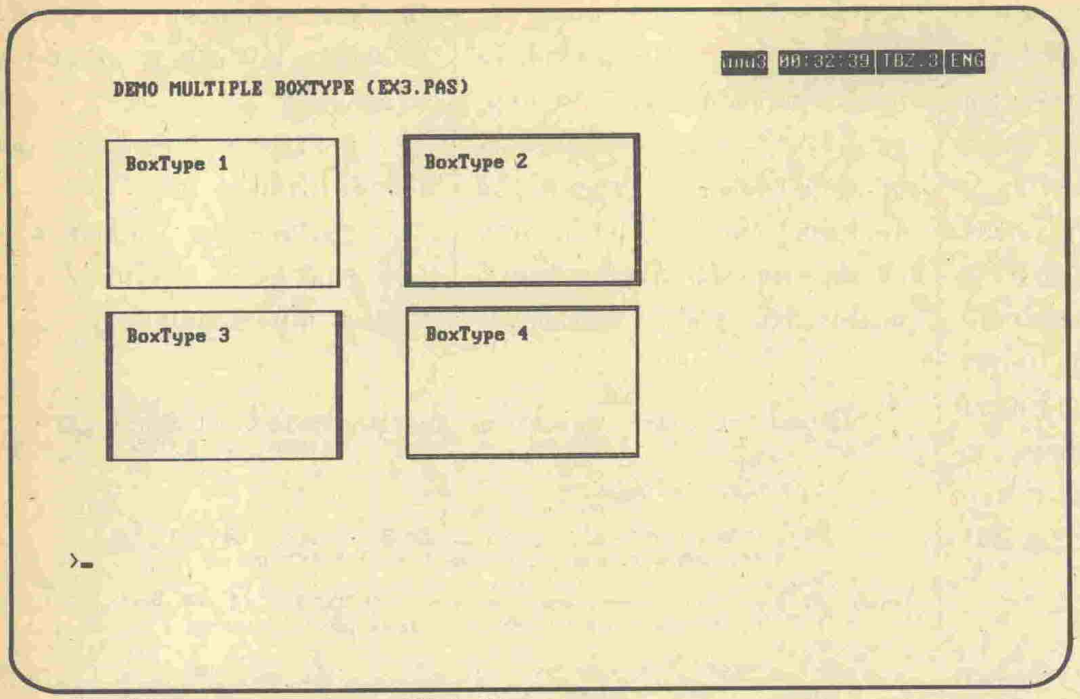
BoxType = 1 กรอบเดี่ยว เช่นเดียวกับโปรซีเจอร์ Box1

BoxType = 2 กรอบคู่ตลอด

BoxType = 3 เส้นบนล่างเป็นเส้นเดี่ยว เส้นข้าง ๆ เป็นเส้นคู่

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  12:29:00                  Listing of EX3.PAS

    1: Program BoxExample;
    2:
    3: { This program illustrate a Standard Pascal way of
    4:   drawing a rectangular box on an IBM-PC screen }
    5:
    6: { ========================== PROCEDURE BOX2 ======= }
    7:   Procedure Box2 (x0,y0,x1,y1,BoxType : integer);
    8:
    9:   const
   10:      fmax     = 4;
   11:   type
   12:      BoxPart   = array [1..fmax] of char;
   13:   const
   14:      { BoxType ==>          $ 1    2    3    4 }
   15:      topLEFT  : BoxPart = (#$da,#$c9,#$d6,#$d5);
   16:      top      : BoxPart = (#$c4,#$cd,#$c4,#$cd);
   17:      topRIGHT : BoxPart = (#$bf,#$bb,#$b7,#$b8);
   18:      botLEFT  : BoxPart = (#$c0,#$c8,#$d3,#$d4);
   19:      bot      : BoxPart = (#$c4,#$cd,#$c4,#$cd);
   20:      botRIGHT : BoxPart = (#$d9,#$bc,#$bd,#$be);
   21:      side     : BoxPart = (#$b3,#$ba,#$ba,#$b3);
   22:   var  i : integer;
   23:        ch: char;
   24:
   25:   begin
   26:      If (BoxType < 1) or (BoxType > 4) then BoxType:=1;
   27:      gotoXY(x0,y0);    write(TopLeft[BoxType]);
   28:      ch := Top[BoxType];
   29:      for i := succ(x0) to  pred(x1) do write(ch);
   30:      write(TopRight[BoxType]);
   31:      ch := side[BoxType];
   32:      for i := succ(y0) to pred(y1) do
   33:         begin
   34:         gotoXY(x0,i);  write(ch);
   35:         gotoXY(x1,i);  write(ch);
   36:         end;
   37:      gotoXY(x0,y1);  write(botLeft[BoxType]);
   38:      ch := bot[BoxType];
   39:      for i := succ(x0) to pred(x1) do write(ch);
   40:      write(botRight[BoxType])
   41:   end;
   42:
   43: { ======================= END PROCEDURE BOX2 ======= }
   44:
   45:
   46: begin  { Example Main program }
   47:   ClrScr;
   48:   TextBackground(1);
   49:   GotoXY(4,2);
   50:   Writeln(' DEMO MULTIPLE BOXTYPE (EX3.PAS) ');
   51:   BOX2( 4, 4,24,10,1);   GotoXY(6,5);   Writeln('BoxType 1');
   52:   BOX2(30, 4,50,10,2);   GotoXY(32,5);  Writeln('BoxType 2');
   53:   BOX2( 4,11,24,17,3);   GotoXY(6,12);  Writeln('BoxType 3');
   54:   BOX2(30,11,50,17,4);   GotoXY(32,12); Writeln('BoxType 4');
   55:   GotoXY(1,20)
   56: end.
```

รูปที่ 4 โปรแกรมวาดกรอบหลายรูปแบบ



รูปที่ 5
ผลลัพธ์จากโปรแกรม
วาดกรอบหลายรูปแบบ

BoxType = 4 เส้นบนล่างเป็นเส้นคู่ เส้นข้าง ๆ เป็นเส้นเดี่ยว

โปรแกรมวาดกรอบแบบสลับซับซ้อน

โปรซีเจอร์ Box2 นับว่าใช้งานได้ดีอยู่เป็นเวลานาน แล้วยังเปิดโอกาสให้เราสามารถเพิ่มเติม BoxType ได้อีกหลายแบบ เช่น การตีเส้นคู่เฉพาะกรอบด้านบน ที่เหลือสามด้านเป็นเส้นเดี่ยว ฯลฯ แต่เมื่อใช้งานมาถึงขนาดหนึ่ง ผู้ใช้ utility routine ในมหาวิทยาลัยธรรมศาสตร์เองพบว่ายังไม่เพียงพอ มีกรอบแบบอื่น ๆ อีกที่น่าจะทำเป็นยูทิลิตี้ด้วย แต่ไม่มีใน Box2 ตัวอย่างความต้องการของโปรแกรมเมอร์ ได้แก่

- ทำกรอบ พร้อมกับตีเส้นคาดด้านบนสำหรับใส่หัวเรื่อง
- ทำให้ยูทิลิตี้รูทีนสามารถวาดกรอบและลบจอภาพส่วนที่อยู่ในกรอบให้ด้วย
- ต้องลบเฉพาะเส้นวาดกรอบได้ด้วยถ้าต้องการ
- คาดเส้นทั้งด้านบนด้านล่างของกรอบ เพื่อใส่หัวเรื่องและหมายเหตุใต้กรอบ

ดูแล้วชวนปวดหัวไม่ใช่น้อย แต่ก็พบว่าโปรซีเจอร์ Box2 จัดได้ว่าเป็นพื้นฐานที่ดีในการขยายโครงสร้างข้อมูล (เพื่อรับกับรูปร่างใหม่ ๆ) และขยายอัลกอริทึม (เพื่อรับกับเงื่อนไขใหม่ ๆ) แม้กระนั้นก็ตาม ความต้องการที่ยกตัวอย่างมาข้างต้นนี้จัดได้ว่ามากเกินกว่าที่ Box2 จะขยายได้ การออกแบบโปรซีเจอร์สำหรับวาดกรอบจึงได้รับการยกเครื่องทั้งหมด โดยเขียนใหม่เป็นภาษาปาสคาลเช่นเดิม แต่ได้ปรับปรุงหลายอย่าง เช่น

1. มีการเรียกใช้โปรซีเจอร์ใน DISPLAY.LIB (อธิบายไว้ในไมโครฯ เล่ม 33) เพื่อการวาดกรอบที่เร็วกว่า Write หลายเท่าตัว

2. มีการผ่านพารามิเตอร์เข้าโปรซีเจอร์ Box โดยเข้าเป็น record ที่มีสรรพคุณของกรอบไว้ครบครัน เรียกชื่อ data type นี้ว่า BoxDesc

3. มี BoxType (ส่วนหนึ่งของ BoxDesc) ให้ถึง 8 แบบ แต่ละแบบสามารถเลือกได้ว่าจะวาดเฉพาะกรอบ (BoxType 1 ถึง 8) หรือวาดกรอบแล้วให้เขียนทับลงในกรอบด้วยตัวอักษร FillerChar/FillerAttr ด้วย (BoxType $81 ถึง $88)

4. เพิ่มกรณีพิเศษ BoxType=0 จะลบทั้งกรอบและภายในกรอบ และกรณี BoxType=$80 ลบเฉพาะกรอบ การลบนี้หมายถึงการเขียนทับด้วยตัวอักษร FillerChar/FillerAttr

ในการใช้งาน BOX.LIB ผู้ใช้จะต้องมีการ include ไฟล์ชื่อ TYPEDEF.LIB และ DISPLAY.LIB ไว้ด้วย รูปที่ 6-8 เป็น listing ของ TYPEDEF.LIB DISPLAY.LIB และ BOX.LIB ตามลำดับ รูปที่ 9 เป็นตัวอย่างโปรแกรมใช้แสดงความสามารถของ BOX ตามที่อธิบายมานี้ทั้งหมด ผลที่ได้จากการใช้ BOX.LIB โดยการรันโปรแกรมในรูปที่ 9 ปรากฏในรูปที่ 10

รูปที่ 6 ไฟล์ Typedef.Lib

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  12:33:00                Listing of TYPEDEF.LIB

    1: { FILE TYPEDEF.LIB  VERSION 1.00
    2:   10/05/87  - This is a preliminary version of TYPEDEF.LIB.
    3:               It is subject to improvements and extensions at any time }
    4:
    5:
    6: type
    7:    str2    = string   [2];
    8:    str4    = string   [4];
    9:    str6    = string   [6];
   10:    str8    = string   [8];
   11:    str10   = string  [10];
   12:    str16   = string  [16];
   13:    str20   = string  [20];
   14:    str40   = string  [40];
   15:    str80   = string  [80];
   16:    str128  = string [128];
   17:    str255  = string [255];
   18:    WorkString = str255;
   19:    Screen  = array[1..25,1..80] of integer;
   20:    LScreen = array[0..1999] of integer;
   21:    BoxDesc = record
   22:                BorderAttr,                 { attribute of the border }
   23:                BoxType,                    { 0, 1..8, $80, $81..$88  }
   24:                W, H          : byte;       { width and height of box }
   25:                FillerChar    : Char;       { Filler character        }
   26:                FillerAttr,                 { Filler attribute        }
   27:                X, Y          : byte        { top left-hand corner    }
   28:              end;
   29:
   30:
   31: var      Vram     : ^Screen;
   32:
   33: { ---------------------------------- end of TYPEDEF.LIB ------------ }
```

รูปที่ 7
ไฟล์ Display.Lib

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  12:35:00                Listing of DISPLAY.LIB

    1: {  DISPLAY.LIB   Version 1.00  08/26/87
    2:    Thaweesak Koanantakool, IPIED/Thammasat University
    3:
    4:    Prerequisites:  Pre-defined types   WorkString, Screen
    5:                    Pre-defined vars    Vram                }
    6:
    7: var
    8:    CrtMode  : byte absolute $40:$49;
    9:
   10:
   11: Procedure DefineVram;
   12:
   13:   { Call to this routine at the beginning of your program
   14:     to get the right address for Monochrome or color card.
   15:     Failing to call to this routine will default to Monochrome. }
   16:
   17:    begin
   18:      If CRTmode=3 then Vram:=Ptr($B800,0) else Vram := Ptr($B000,0)
   19:      end;  { DEFINE VRAM }
   20:                                                          |
   21:
   22: Procedure PutFast( c:char; attribute:byte; x,y : integer);
   23:    begin
   24:      Vram^[y,x] := swap(attribute) + ord(c)
   25:      end;  { PUTFAST }
   26:
   27:
   28: Procedure WriteFast( Msg : WorkString; attribute:byte;  X,Y : integer);
   29:    var
   30:       a,i,j : integer;
   31:    begin
   32:       a := swap(attribute);
   33:       For i:= 1 to Length(Msg) do Vram^[y,pred(x+i)]:=a + ord(Msg[i]);
   34:       end; {  WRITEFAST }
   35:
   36:
   37: {---------------------------------------- end of DISPLAY.LIB ------- }
   38:
```

รูปที่ 8 ไฟล์ BOX.Lib สำหรับใช้เป็นยูทิลิตี้วาดกรอบแบบสลับซับซ้อน

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  12:58:00                  Listing of BOX.LIB

    1: { FILE BOX.LIB 09/19/87 (c) 1987 Thaweesak Koanantakool }
    2: { Prerequisites:  TYPEDEF.LIB
    3:                   DISPLAY.LIB }
    4: (* BoxDesc = record
    5:                BorderAttr,                { attribute of the border }
    6:                BoxType,                   { 0, 1..8, $80, $81..$88  }
    7:                W, H          : byte;      { width and height of box }
    8:                FillerChar    : Char;      { Filler character        }
    9:                FillerAttr,                { Filler attribute        }
   10:                X, Y          : byte       { top left-hand corner    }
   11:              end;
   12:
   13:   Specifications of BoxType :
   14:           1  : special Box
   15:           2  : single line border
   16:           3  : double line border
   17:           4  : single line border with double line sides
   18:           5  : single line border with double line top/bottom
   19:           6  : single line border with single line seperate header
   20:           7  : double line border with double line seperate header
   21:           8  : double line border with single line seperate header
   22:    If BoxType has the most significant bit set (i.e. $81..$88)
   23:    then the inside of the box shall be filled with the character
   24:    FillerChar with attribute FillerAttr.
   25:    If BoxType = 0 then the whole area shall be cleared.
   26:    If BoxType = $80 border shall be cleared
   27:                                                              *)
   28:
   29: Function DupChar( c : char;  n : integer) : Str255;
   30: var    I : integer;
   31: Begin  For i:=1 to n do DupChar[i]:=c;
   32:       DupChar[0]:=chr(n)
   33: end;
   34:
   35:
   36: Procedure Box ( BoxRec : BoxDesc );
   37: const  fmax     = 8;
   38: type   BoxPart   = array [0..fmax] of char;
   39: const
   40:    { BoxType ==>          $80     1    2    3    4    5    6    7    8 }
   41:    topLEFT  : BoxPart = (#$20,#$d5,#$da,#$c9,#$d6,#$d5,#$da,#$c9,#$c9);
   42:    top      : BoxPart = (#$20,#$cd,#$c4,#$cd,#$c4,#$cd,#$c4,#$cd,#$cd);
   43:    topRIGHT : BoxPart = (#$20,#$b8,#$bf,#$bb,#$b7,#$b8,#$bf,#$bb,#$bb);
   44:    botLEFT  : BoxPart = (#$20,#$c0,#$c0,#$c8,#$d3,#$d4,#$c0,#$c8,#$c8);
   45:    bot      : BoxPart = (#$20,#$c4,#$c4,#$cd,#$c4,#$cd,#$c4,#$cd,#$cd);
   46:    botRIGHT : BoxPart = (#$20,#$d9,#$d9,#$bc,#$bd,#$be,#$d9,#$bc,#$bc);
   47:    side     : BoxPart = (#$20,#$b3,#$b3,#$ba,#$ba,#$b3,#$b3,#$ba,#$ba);
   48:    midLEFT  : BoxPart = (#0  ,#$c6,#0,  #0,  #0,  #0,  #$c3,#$cc,#$c7);
   49:    midLINE  : BoxPart = (#0  ,#$cd,#0,  #0,  #0,  #0,  #$c4,#$cd,#$c4);
   50:    midRIGHT : BoxPart = (#0  ,#$b5,#0,  #0,  #0,  #0,  #$b4,#$b9,#$b6);
   51:    lowLEFT  : BoxPart = (#0  ,#$c3,#0,  #0,  #0,  #0,  #0  ,#0  ,#0  );
   52:    lowLINE  : BoxPart = (#0  ,#$c4,#0,  #0,  #0,  #0,  #0  ,#0  ,#0  );
   53:    lowRIGHT : BoxPart = (#0  ,#$b4,#0,  #0,  #0,  #0,  #0  ,#0  ,#0  );
   54:
   55: var   Xpos, Ypos, Xl, Yl  : Byte;
   56:
   57:
   58:    Procedure ClipBox;              { check that box is on screen }
```

รูปที่ 8 (ต่อ)

```
59:   begin
60:   with BoxRec do
61:      begin
62:      X1 := X + W - 1;
63:      Y1 := Y + H - 1;
64:      if X < 1 then X := 1;
65:      if Y < 1 then Y := 1;
66:      if X1 > 80  then begin X1 := 80 ; W := 81 - X end;
67:      if Y1 > 25  then begin Y1 := 25;  H := 26 - Y end;
68:      end   { with }
69:   end; { ClipBox }
70:
71:   Procedure FillBox;      { fill inside of the box for BoxType 0,$80..$88 }
72:   begin
73:   with BoxRec do
74:      begin
75:      for Ypos := Y to Y1 do
76:          Writefast (DupChar(FillerChar,W), FillerAttr, X, Ypos);
77:      end;  { with }
78:   end;  { FillBox }
79:
80:   procedure SpecialBox;
81:   begin
82:   with BoxRec do
83:      begin
84:      WriteFast (midLEFT [BoxType] + DupChar (midLINE [BoxType], W - 2) +
85:                midRIGHT [BoxType],
86:                BorderAttr, X, Y + 2);
87:      If BoxType=1 then
88:      WriteFast (lowLEFT [BoxType] + DupChar (lowLINE [BoxType], W - 2) +
89:                lowRIGHT [BoxType],
90:                BorderAttr, X, Y1 -2);
91:      end;  { with }
92:   end; { SpecialBox }
93:
94:   begin { Box }
95:   ClipBox;
96:   with BoxRec do
97:      begin
98:      if (BoxType and $7F) > fmax then exit;
99:      If BoxType = 0 then begin FillBox; exit end;
100:     If BoxType - $80 > 0 then FillBox;
101:     BoxType:=BoxType and $7F;
102:     { draw top line }
103:     WriteFast (topLEFT [BoxType] + DupChar (top [BoxType], W - 2) +
104:               topRIGHT [BoxType],
105:               BorderAttr, X, Y);
106:     { draw side lines }
107:     for Ypos := succ (Y) to (Y1 - 1) do
108:        begin
109:        PutFast (side [BoxType], BorderAttr, X, Ypos);
110:        PutFast (side [BoxType], BorderAttr, X1, Ypos)
111:        end;
112:     { draw bottom line }
113:     WriteFast (botLEFT [BoxType] + DupChar (bot [BoxType], W - 2) +
114:               botRIGHT [BoxType],
115:               BorderAttr, X, Y1);
116:     { draw some special line }
117:     if MidLeft[BoxType] <> #0 then SpecialBox
118:     end
119:  end; { Box }
```

ในการเปลี่ยนแปลง procedure parameter จากแบบเดิมคือ มี x0, y0, x1, y1, BoxType มาเป็นแบบ record ประเภท BoxDesc นั้น ผู้เขียนพบว่ามีประโยชน์หลายประการคือ จำนวนรายชื่อที่อยู่ในรายการ parameter เหลือเพียงชื่อเดียวดังที่เห็นได้จากรูปที่ 9 บรรทัดที่ 40 นอกจากนี้ในการกำหนดความหมายของ MyBox ในบรรทัดที่ 21-24 ก็ช่วยให้เราเขียนโปรแกรมในลักษณะที่อ่านง่าย ทั้งนี้เพราะรายละเอียดทุกอย่างของ MyBox จะถูกรวบรวมไว้ด้วยกันไม่กระจัดกระจาย เหตุผลอันสุดท้ายก็คือความเร็วในการทำงานและความยาวของโปรแกรมหลักที่เรียกใช้ Box จะดีทั้งคู่ ทั้งนี้เพราะตัวพารามิเตอร์ที่ส่งให้โปรซีเจอร์ต่าง ๆ ปาสคาลส่งทางสแต็ก ยิ่งมีมากชื่อยิ่งต้อง push ลงสแต็กกันมาก แต่ถ้าลดลงเหลือเป็นประเภท var เพียงตัวเดียว (ดูรูปที่ 8 บรรทัดที่ 40) การ push พารามิเตอร์จะเหลือเพียง 2 word ซึ่งจัดได้ว่าเป็นวิธีการที่กะทัดรัดที่สุดเท่าที่จะทำได้แล้ว

แม้การ call โปรซีเจอร์จะสั้น แต่

รูปที่ 9
โปรแกรมสำหรับแสดงความสามารถของ BOX.Lib

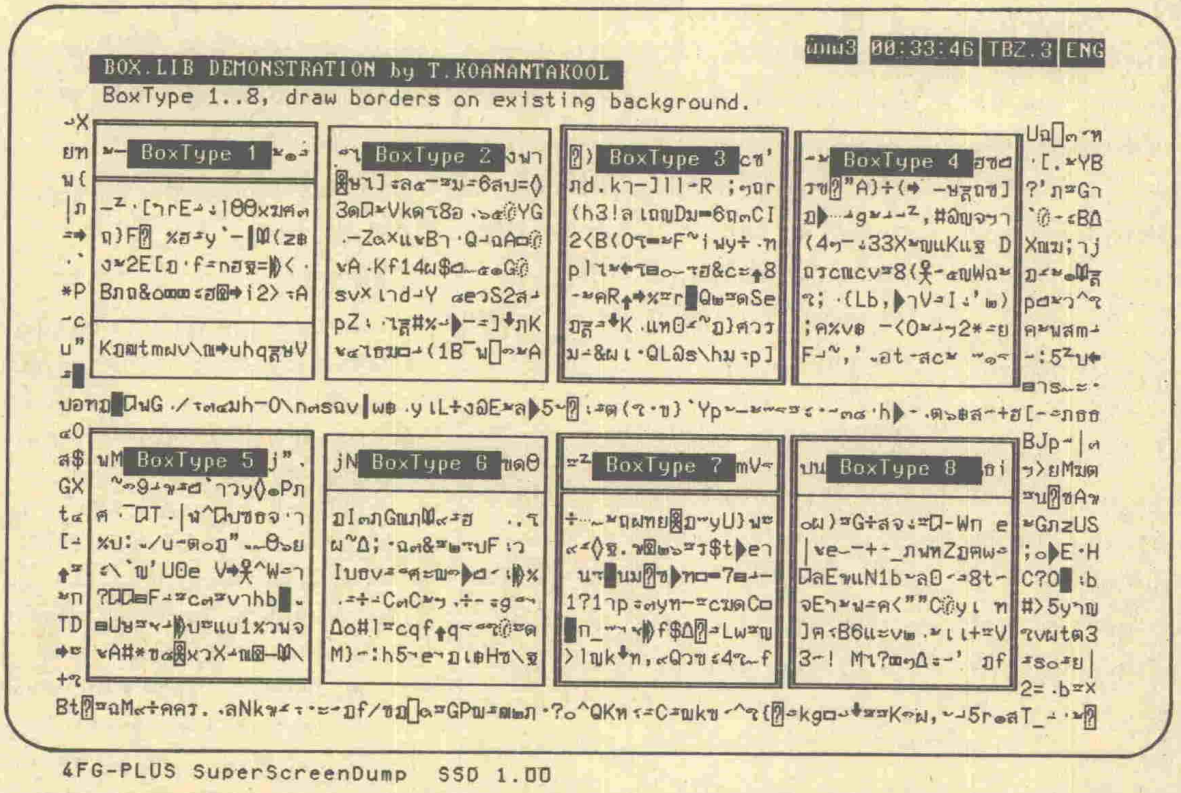
```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  13:00:00                 Listing of EX4.PAS

    1: Program BoxDemo;
    2:
    3: {$I TYPEDEF.LIB }
    4: {$I DISPLAY.LIB }
    5: {$I BOX.LIB }
    6:
    7: procedure FillScreen;
    8:   { This procedure fills part of screen with random
    9:     data so that you may see the effect of BOX() }
   10: var  xx,yy : integer;
   11: begin
   12:    for yy := 4 to 25 do
   13:      for xx := 1 to 80 do
   14:           PutFast(chr(Random(255)),$07,xx,yy);
   15: end;
   16:
   17: procedure TestBox(Mode80 : byte);
   18: var
   19:    I  : integer;
   20:    Ch : char;
   21: const
   22:    MyBox : BoxDesc
   23:          = (BorderAttr : $1F; BoxType:1; W:18; H:10;
   24:             FillerChar:'+'; FillerAttr:$37; X:10; Y:3);
   25: begin
   26:    GotoXY(3,2);   TextBackground(14);
   27:    Writeln(' BOX.LIB DEMONSTRATION by T.KOANANTAKOOL ');
   28:    TextBackground(0);
   29:    If Mode80=0
   30:       then writeln('   BoxType 1..8, draw borders on existing background. ')
   31:       else Writeln('   BoxType $81..$88, box filled by ',
   32:                    'character of your choice. ');
   33:    TextBackground(13);
   34:    for I := 1 to 8 do
   35:       With MyBox do
   36:         begin
   37:         X := 3+18*(pred(I) mod 4);
   38:         Y := 4+11*(pred(I) div 4);
   39:         BoxType := I+Mode80;
   40:         Box (MyBox);
   41:         gotoXY (X+3,Y+1);  write(' BoxType ',BoxType,' ');
   42:         end;  { with/ for }
   43:    TextBackground(0);
   44:    end; { TestBox }
   45:
   46: begin
   47:    DefineVRAM;
   48:    ClrScr;    FillScreen;
   49:    { Draw boxes without clearing texts within box area }
   50:    TestBox(0);
   51:    Readln;    FillScreen;
   52:    { Draw boxes with filler characters }
   53:    TestBox($80);
   54:    Readln
   55: end.
```

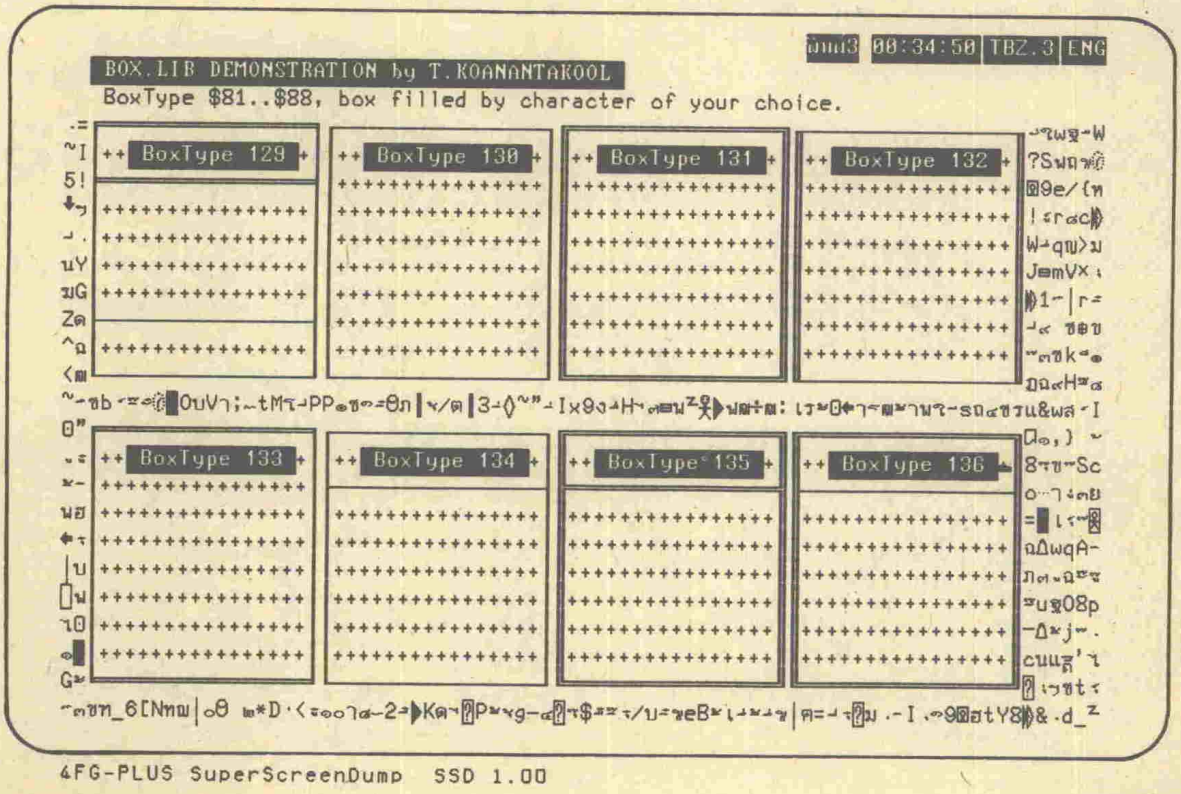
เนื่องจากตัวแปรประเภท BoxDesc มีรายละเอียดค่อนข้างมาก การทำงานจึงสามารถทำได้ครบถ้วนตามความต้องการ คำจำกัดความของ BoxDesc (ภาษาปาสคาล) ปรากฏในรูปที่ 6 (บรรทัดที่ 21 ถึง 28) เพื่อช่วยให้เกิดความชัดเจนมากขึ้น จะขออธิบายเพิ่มเติมดังนี้

BorderAttr คือ แอตทริบิวต์ไบต์สำหรับตัวอักษรที่ใช้ทำกรอบ

BoxType คือ ตัวเลขบอกประเภทของกรอบ (0 ถึง 8 และ $80 ถึง $88)

W คือ ความกว้างของกรอบ นับ





รูปที่ 10 ผลลัพธ์ของการใช้ BOX.LIB หรือ BOX4FG.LIB ผ่านโปรแกรม EX4 เครื่องหมาย + จะโปรแกรมให้เป็น blank หรือตัวอักษรอะไรก็ได้

เป็นตัวอักษร

H คือ ความสูงของกรอบ นับเป็นบรรทัดตัวอักษร

FillerChar คือ ตัวอักษรที่จะใช้เติมลงในกรอบ ในกรณีที่ BoxType เป็น $81 ถึง $88

FillerAttr คือ แอตทริบิวต์ไบต์สำหรับตัวอักษรที่จะใช้เติมลงในกรอบ

X คือ ตำแหน่งทางแนวนอนของมุมซ้ายบนของกรอบ

Y คือ ตำแหน่งทางแนวดิ่ง ของมุมซ้ายบนของกรอบ

โปรแกรมสลับซับซ้อนแต่ใช้ง่ายมาก

โปรซีเจอร์ BOX ในรูปที่ 8 จัดได้ว่ายาวพอสมควร และเมื่อคอมพายล์มาแล้ว เฉพาะส่วนนี้กินเนื้อที่ code ประมาณ 1360 ไบต์ เมื่อนำไปใช้ร่วมกับโปรแกรมขนาดใหญ่อาจประสบปัญหาเรื่อง code segment เกิน 64K และเสียเวลาคอมพายล์ส่วนของ BOX .LIB ผู้เขียนและคณะจึงได้ทำการเขียน BOX เสียใหม่ เป็นภาษาแอสเซมบลี รวมไว้เป็นส่วนหนึ่งของยูทิลิตี้รูทีนสำหรับการประมวลผลภาษาไทยความเร็วสูง โดยให้เป็นฟังก์ชันบริการหมายเลข 45h ของอินเตอร์รัพต์ 10h ตัวอย่างการเรียกใช้โปรซีเจอร์ BOX4FG ซึ่งมีคุณสมบัติเหมือนกับ BOX.LIB ทุกประการ แต่ยาวเพียง 112 ไบต์ แสดงไว้ในรูปที่ 11 พร้อมกับโปรแกรมทดสอบในรูปที่ 9 (เพียงแก้บรรทัดที่ 5 จาก BOX.LIB เป็น BOX4FG.LIB) ซึ่งทดลองได้กับไทยการ์ด 4FG และระบบภาษาไทย 4FG.COM รุ่น 1.20 หรือสูงกว่านี้

สรุป

การออกแบบโปรแกรมจัดได้ว่าเป็นเทคนิคสำคัญที่จะช่วยให้โปรแกรมที่พัฒนาขึ้นมีความสามารถสูง ปรับตัวง่าย ทำงานเร็วและสั้นได้ในเวลาเดียวกัน การออกแบบโปรแกรมอย่างระมัดระวังจะช่วยให้สามารถพัฒนาโปรแกรมได้เร็ว ทั้งนี้ต้องทำให้อัลกอริทึมง่ายเอาไว้ก่อน ในขณะเดียวกันการเลือกใช้โครงสร้างข้อมูลที่ดีก็จะช่วยให้การขยายความสามารถของโปรแกรมเป็นไปได้ง่ายและสะดวก และในที่สุดหากรูทีนไหนที่ใช้กันบ่อยเมื่อเขียนเป็นแอสเซมบลีให้ใช้กันแล้ว โปรแกรมเมอร์จะได้ประโยชน์มากขึ้นอีก เพราะช่วยให้งานเร็วขึ้น ซอร์สโค้ดสั้นลง โปรแกรมที่คอมพายล์ออกมาก็เล็กลง แถมยังทำงานเร็วกว่าเดิมมาก

โปรดติดตามตอนต่อไปในวารสารไมโครฯ ฉบับหน้า

หมายเหตุ รูปที่ 7, 12 เป็นโปรแกรม-**EX1.PAS** ซึ่งเป็นตัวอย่างโปรแกรมสำหรับบทความนี้ในไมโครฯ เล่ม 33 ซึ่งได้แจกเป็นใบแทรก ทางผู้เขียนและกองบก.ขออภัยที่ได้ทำ Listing นี้ตกหล่นไป

ผู้เขียนขอถือโอกาสนี้ขอบคุณวิศวกรซอฟต์แวร์ของบริษัท เทลบิซ จำกัด ที่มีส่วนสำคัญในการพัฒนาโปรแกรมวาดกรอบ คุณวิวัฒน์ วงศ์โสพนากุล ในการเขียนโปรแกรม Box เป็นภาษาแอสเซมบลีเพื่อใช้ในโครงการนี้ คุณสันติ ยอดเพชร ในการแสดงภาพถ่ายของจอภาพออกทางเครื่องพิมพ์ดอตแมทริกซ์ คุณสถาพร ก่อรักเศวต ในการออกแบบข้อกำหนดของ BOX.LIB

รูปที่ 11
ไฟล์ BOX4FG.LIB ใช้แทน box.lib เพื่อวาดกรอบแบบสลับซับซ้อน โดยเรียกใช้บริการจากระบบภาษาไทย 4 FG รุ่น 1.20 หรือ 4 FG-Plus

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  13:01:00                  Listing of BOX4FG.LIB

    1: { FILE BOX4FG.LIB 09/19/87 (c) 1987 Thaweesak Koanantakool }
    2:
    3: { Prerequisites  :  TYPEDEF.LIB
    4:                     DISPLAY.LIB }
    5:
    6: { Specifications : identical to BOX.LIB (approx 1360 bytes) written in Pascal }
    7: { Availability   : with TELBIZ/4FG.COM version 1.20 or higher }
    8: { CodeSize       : BOX4FG.LIB adds approx. 112 bytes to your code }
    9:
   10: Procedure Box ( var BoxRec : BoxDesc );
   11: const
   12:    VaBBoxWHBAXY = $457A;    { 4FG Int 10h function for drawing a box }
   13:
   14: begin
   15:    with BoxRec do
   16:    begin
   17:    x:=pred(x); y:=pred(y);              { converts PascalXY to BIOS XY }
   18:    inline ( $1E/                        { PUSH  DS  }
   19:             $C5/$76/<BoxRec/            { LDS   SI,[BP].BoxRec }
   20:             $b8/VABBoxWHBAXY/           { MOV   AX,$457A }
   21:             $CD/$10/                    { INT   10H     }
   22:             $1F);                       { POP   DS      }
   23:    x := succ(x); y:=succ(y)             { restores value of X/Y }
   24:    end;  { with }
   25: end;
```

รูปที่ 12 ไฟล์ EX1.PAS สำหรับแสดงความสามารถของ Display.Lib

```
Thammasat University  TurboPascal Ver 3.01A  File Listing Utility
page 1    10/05/87  13:02:00                  Listing of EX1.PAS

    1: PROGRAM DIRECT_VIDEO_WRITE;
    2:
    3: { Purpose: To demonstrate the speed difference between Write/Writeln
    4:   statements of TurboPascal and direct-video write to screen.
    5:   Note that Write/Writeln is a loop consisting of the following operations:
    6:
    7:    repeat for every character to be shown on screen
    8:      * set cursor position (Int 10h)
    9:      * write character/attribute at current position (Int 10h)
   10:
   11:   Thaweesak Koanantakool, IPIED  Thammasat University,  March, 1987  }
   12:
   13: type
   14:    WorkString  = string[255];
   15:    Screen    = array[1..25,1..80] of integer;
   16:
   17: const
   18:    VramPtr   : array [1..2] of integer = ($0000,$b000);
   19: var
   20:   Vram      : ^Screen absolute VramPtr;
   21:   TestString : WorkString;
   22:
   23: {$I DISPLAY.LIB }
   24:
   25: Procedure Test;
   26: var L : integer;
   27:
   28:      Procedure Prompt (Msg : WorkString);
   29:        begin  { PROMPT }
   30:        GotoXY(1,25); Write(Msg);  Readln; ClrScr
   31:        end;   { PROMPT }
   32:
   33:      Procedure BIOSTest;
   34:        begin
   35:        for L := 1 to 24 do
   36:          begin
   37:          gotoXY(1,L);
   38:          write(TestString+'BIOS MODE')
   39:          end;
   40:        end;  { BIOSTEST }
   41:
   42:     Procedure DirectVideoTest;
   43:        begin                          { attribute $07 = normal video }
   44:        for L := 1 to 24 do WriteFast ('DIRECT '+TestString, $07, 1,L);
   45:        end;  { DIRECTVIDEOTEST }
   46:
   47:      begin  { TEST }
   48:      ClrScr;
   49:      DefineVram;
   50:      TestString := 'This is a demonstration to compare the'+
   51:                    ' speed of BIOS and direct video..';
   52:      Prompt('Press RETURN to start BIOS display.');
   53:      BIOSTest;
   54:      Prompt('Press RETURN to start Direct video test. ');
   55:      DirectVideoTest;
   56:      Prompt('Press RETURN to end program.');
   57:      end;   { TEST }
   58:
   59: begin  { MAIN PROGRAM }
   60:      Test
   61: end.   { MAIN PROGRAM }
```